Module 5

Ananya Verma, Vikas Jadhav Y

March 2025

Activity 1

Implementing a Perceptron from Scratch

Input Representation

The perceptron starts with an input vector:

$$a^{(0)} = X$$

Each hidden layer computes a weighted sum of its inputs using weight matrices $\mathbf{W}$ and bias vectors $\mathbf{b}$, given by:

$$z^{(i)} = \mathbf{W}^{(i)} a^{(i)} + \mathbf{b}^{(i)}, \quad \forall i \in \{0, \ldots, L-1\}$$

This is then passed through an activation function.

Activation Functions

The activation function differs based on the type of task:

For Regression:

$$a^{(i)} = \begin{cases} \tanh(z^{(i)}), & \text{if } i < L-1 \text{ (hidden layers)} \\ z^{(i)}, & \text{if } i = L-1 \text{ (output layer)} \end{cases}$$

For Classification:

$$a^{(i)} = \begin{cases} \tanh(z^{(i)}), & \text{if } i < L-1 \text{ (hidden layers)} \\ \sigma(z^{(i)}), & \text{if } i = L-1 \text{ (output layer)} \end{cases}$$

where the **sigmoid function** is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

At the final layer, the predicted output is:

$$y_{\text{pred}} = a^{(L-1)}$$

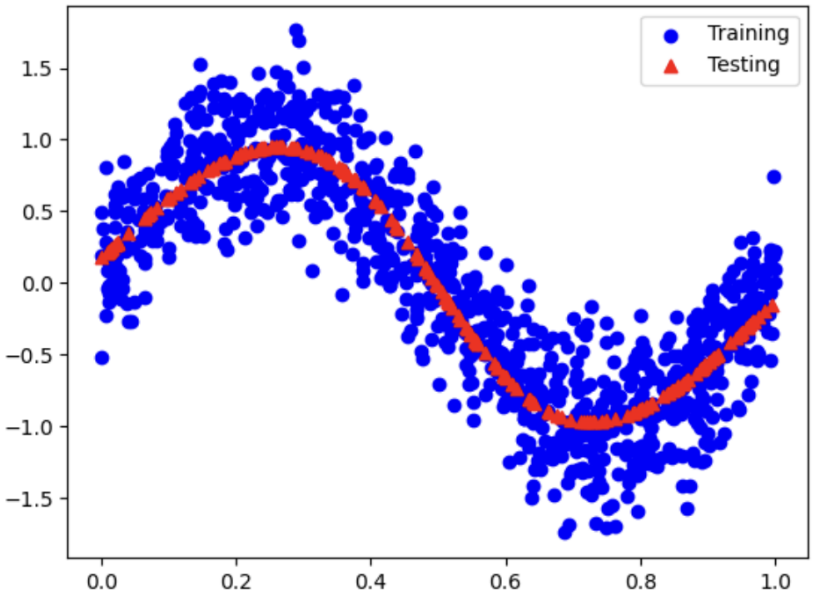

Figure 1: Training and testing data

Training the Perceptron

During training, the weights and biases are updated iteratively over multiple epochs using the following update rules:

$$\mathbf{W}^{(i)} \leftarrow \mathbf{W}^{(i)} - \eta \cdot \frac{\partial L}{\partial \mathbf{W}^{(i)}}$$

$$\mathbf{b}^{(i)} \leftarrow \mathbf{b}^{(i)} - \eta \cdot \frac{\partial L}{\partial \mathbf{b}^{(i)}}$$

where η represents the **learning rate**.
The gradients are computed as:

$$\frac{\partial L}{\partial \mathbf{W}^{(i)}} = \frac{1}{m}\left(\frac{\partial L}{\partial z^{(i)}} a^{(i-1)T}\right)$$

$$\frac{\partial L}{\partial \mathbf{b}^{(i)}} = \frac{1}{m}\sum \frac{\partial L}{\partial z^{(i)}}$$

The term $\frac{\partial L}{\partial z^{(i)}}$ depends on the derivative of the activation function used, which varies depending on whether the task is regression or classification. The regression using the test data from the sine plus noise data is given in Fig 1 and the decision boundary for the classification problem from last module is shown in Fig 2.

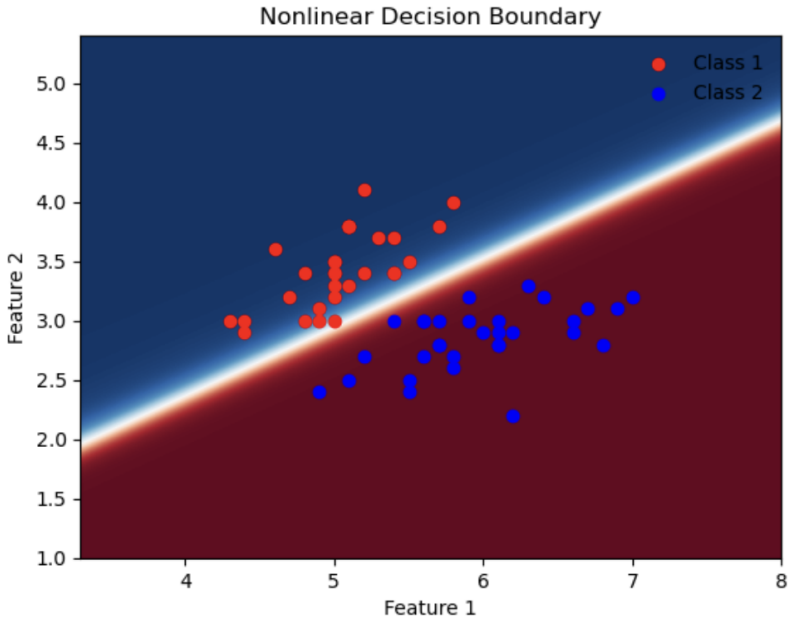

Figure 2: Decision Boundary

Activity 2

Exploring Parameter Variations

We experimented with different parameter settings for various dataset sizes. Our observations indicate that increasing the number of neurons in a single hidden layer enhances the **R^2 score**, bringing it closer to 1. Additionally, increasing the number of hidden layers while keeping the same number of neurons per layer also improves the **R^2 score**.

The **R^2 score** was computed using `scikit-learn` for 1000 data points. However, excessively increasing either the number of neurons per layer or the number of hidden layers leads to **overfitting**. This effect becomes particularly noticeable when a large number of hidden layers is used, as depicted in Figures 3 and 4.

For visualization, we plotted results for a subset of 10 data points using a **multi-layer perceptron (MLP)** implemented from scratch.

Furthermore, increasing the number of epochs improved the **R^2 score** and the overall fit. However, due to memory constraints, we were unable to exceed **50,000 epochs** on our system.

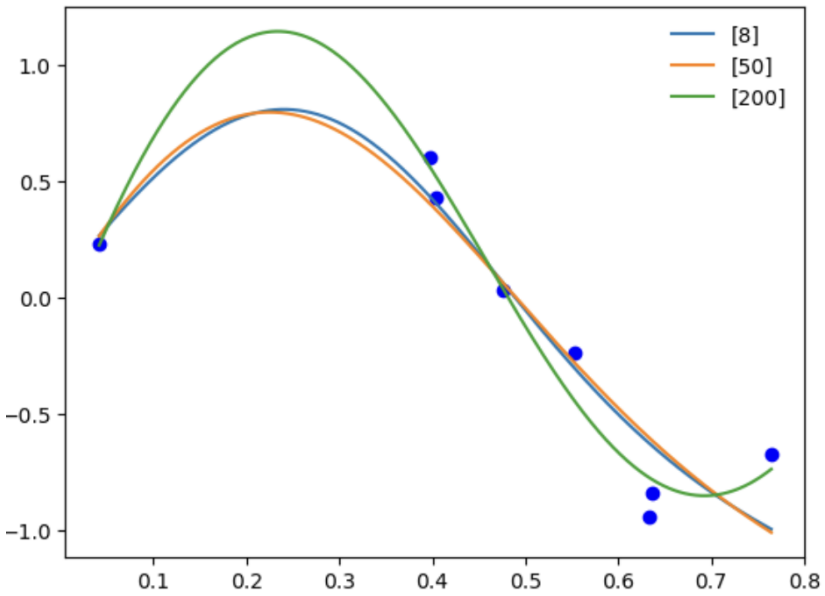

Figure 3: Effect of hidden layer size on regression performance.

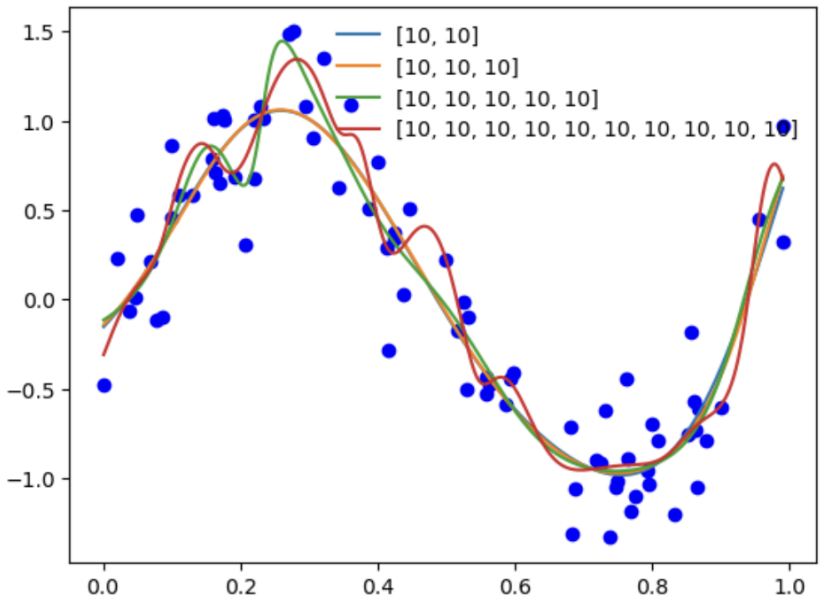

Figure 4: Effect of increasing hidden layers on regression performance.

Activity 3

IRIS Data Classification

The IRIS dataset contains measurements of two flower species: Setosa and Versicolor. We use two features, petal length and sepal length, which are stored in the matrix:

$$X = (x_1, x_2)_{n \times 2}$$

where n represents the number of samples in each class (Class-1: Iris-setosa, Class-2: Iris-versicolor). The class labels, Y_{data}, are assigned as $+1$ for Class-1 and -1 for Class-2.

A 2-layer Perceptron algorithm is trained to predict Y_{predict} for each training sample. The trained model is then used to classify the test dataset. When comparing the predicted test labels $Y_{\text{test, predict}}$ with the actual test labels $Y_{\text{test, data}}$, we observed only two misclassifications.

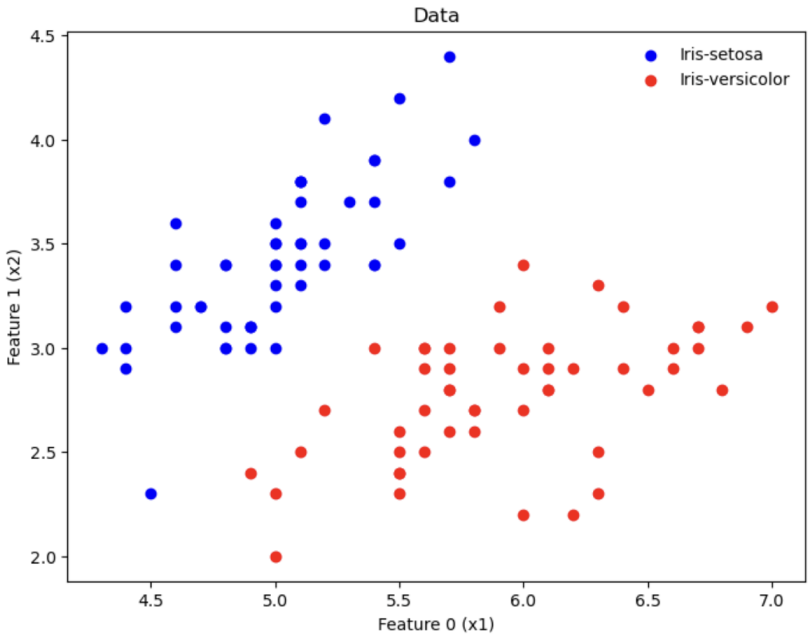

Figure 5: Visualization of the data.